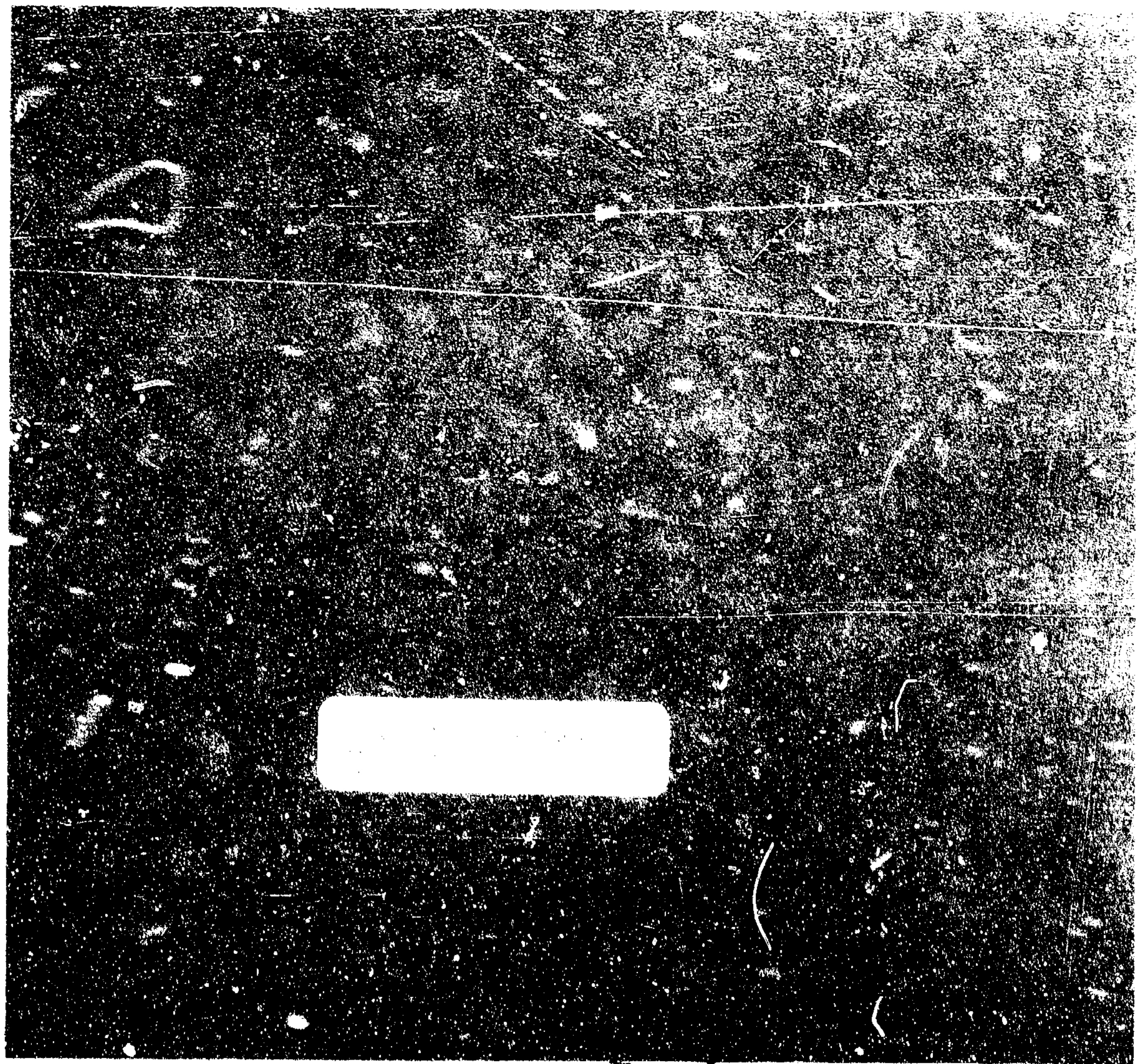
REPORT R-456 FEBRUARY, 1970 UILU-ENG. 70-201

CSL COORDINATED SCIENCE LABORATORY

DESIGN CONSIDERATIONS OF ON-LINE DOCUMENT RETRIEVAL SYSTEMS



UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

37

DESIGN CONSIDERATIONS OF ON-LINE DOCUMENT RETRIEVAL SYSTEMS

BY

J. T. Cordaro Jr. and R. T. Chien
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign

This work was supported in part by the Joint Services Electronics Program (U.S. Army, U.S. Navy, U.S. Air Force) under contract DAAB 07-67-C-0199; by the International Business Machines Corporation; by the National Science Foundation under Grant GK-2339 and by the Office of Education under Grant OE C-1-7-071213-4557.

CSL Reports (continued)

☐ 14. Lum, Vincent and Chien, R. T., "On the Minimum Distance of Bose-Chaudhuri-Hocquenghem Codes," November, 1966, R-328.

☐ 15. Hsu, Hsung Tsao, "Error Correcting Codes for Compound Channels," December, 1966, R-331.

☐ 16. Hong, Se June, "On Minimum Distance of Multiple Error Correcting Arithmetic Codes," January, 1967, R-336.

☐ 17. Gaddess, Terry G., "A Study of an Error Detecting Parallel Adder," January, 1967, M.S. Thesis, R-337.

☐ 18. Preparata, Franco P., "Binary Sequence Convolutional Mapping: The Channel Capacity of a Non-Feedback Decoding Scheme, March, 1967, R-345.

☐ 19. Preparata, F. P. and Chien, R. T., "On Clustering Techniques of Citation Graphs," May, 1967, R-349.

☐ 20. Lipovski, Gerald J., "Compatibility and Row-Column Minimization of Sequential Machines," May, 1967, R-355.

☐ 21. Lipovski, Gerald J., "An Improved Method of Finding all Largest Combinable Classes," August, 1967, R-362.

☐ 22. Chow, David K., "A Geometric Approach to Coding Theory with Application to Information Retrieval," October, 1967, R-368.

☐ 23. Tracey, Robert J., "Lattice Coding for Continuous Channels," December, 1967, R-371.

☐ 24. Preparata, Franco P., "A Study of Nordstrom-Robinson Optimum Code," April, 1968. R-375.

☐ 25. Preparata, Franco P., "A Class of Optimum Nonlinear Double-Error-Correcting Codes," July, 1968, R-389.

☐ 26. Kisylia, Andrew Philip, "An Associative Processor for Information Retrieval," August, 1968, R-390.

☐ 27. Beach, Edward J., "A Study of a Feedback Time-Sharing System," September, 1968, R-391.

CSL Reports (continued)

☐ 28. Weston, P., and Taylor, S. M., "Cylinders: A Data Structure Concept Based on Rings," September, 1968, R-393.

☐ 29. Bahl, Lalit Rai, "Correction of Single and Multiple Bursts of Error," October, 1968, R-397.

☐ 30. Carroll, D. E., Chien, R. T., Kelley, K. C., Preparata, F. P., Reynolds, P., Ray, S. R. and Stahl, F. A., "An Interactive Document Retrieval System," December, 1968, R-398.

☐ 31. Biss, Kenneth, "Syntactic Analysis for the R2 System," December, 1968, R-399.

☐ 32. Tzeng, Kenneth Kai Ming, "On Iterative Decoding of BCH Codes and Decoding Beyond the BCH Bound," January, 1969, R-404.

☐ 33. Kelley, K. C., Ray, S. R. and Stahl, F. A., "ISL-A String Manipulating Language," February, 1969, R-407.

☐ 34. Lombardi, Daniel Joseph, "Context Modeling in a Cognitive Memory," February, 1969, R-408.

☐ 35. Chien, R. T., Hong, S. J. and Preparata, F. P., "Some Results in the Theory of Arithmetic Codes," May, 1969, R-417.

☐ 36. Hong, Se June, "On Bounds and Implementation of Arithmetic Codes," October, 1969, R-437.

☐ 37. Chien, R. T., and Hong, S. J., "Error Correction in High Speed Arithmetic," October, 1969, R-438.

☐ 38. Chien, R. T. and Hong, S. J., "On a Root-Distance Relation for Arithmetic Codes," October, 1969, R-440.

☐ 39. Chien, R. T., "Recent Developments in Algebraic Decoding," November, 1969, R-441.

☐ 40. Chien, Robert T. and Hong, S. J., "An Iterative Approach for the Correction of Iterative Errors," November, 1969, R-443.

CSL Reports (continued)

☐ 41. Jansen, Jr., James Merritt, "Phrase Dictionary Construction Methods for the R2 Information Retrieval Systems," December, 1969, R-447.

☐ 42. Chien, R. T., Ray, S. R., and Stahl, F., "ISL-A New Programming Language for Information Retrieval," December, 1969, R-449.

For copies of these reports please complete this form and send to:

Professor R. T. Chien
Coordinated Science Laboratory
University of Illinois
Urbana, Illinois 61801

My name and address is ______________________________
____
____
____

DESIGN CONSIDERATIONS OF ON-LINE DOCUMENT RETRIEVAL SYSTEMS*

J. T. Cordaro Jr. and R. T. Chien

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign

Abstract

Response times of on-line document retrieval systems are analyzed. Linear and inverted file organizations are considered and their response times are evaluated.

*This work is supported in part by the Joint Services Electronics Program (U.S. Army, U.S. Navy, and U.S. Air Force) under contract DAAB-07-67-C-0199, by the International Business Machines Corporation, by the National Science Foundation under Grant GK-2339 and by the Office of Education under Grant OE C-1-7-071213-4557.

DESIGN CONSIDERATIONS OF ON-LINE DOCUMENT RETRIEVAL SYSTEMS

I. Introduction

The increasing population of information centers and computer systems with remote terminals provides the necessary ingredients for the development of real-time information systems. Such a system has many advantages over conventional systems.

The most important advantage is obviously the time factor. While saving of time is of importance to all users, in many cases it spells the difference between success and failure of a mission or induces the adoptation of less attractive alternatives. Another major advantage of real-time remote-access systems is better utilization and sharing of resources, hence the reduction of the high cost of maintaining local information stores. Library networks are good examples of this.

Conventional document retrieval systems are batch processed. The response time runs into hours or even days. These systems are mostly tape oriented with human interface. Real-time on-line systems are not only fast but also can be organized to improve the efficiency of the system. The response time should be in seconds or at most minutes.

Current document retrieval systems are often run in off hours in batch. This would be undesirable for on-line systems. To attain the economical objective the on-line system could either be part of a time-sharing system or have a large number of terminals, and therefore users. Either solution would put severe restrictions on the response time of the system. The purpose of this paper is to evaluate the performance, particularly response time, of a system totally devoted to the purpose of document retrieval. Two organizations and search strategies are evaluated. We also present a method of processing which appears to give for linear files the best response time with a given number of terminals. This new method utilizes a two level search organization.

The organization of this paper is as follows. A discussion of basic concepts in file organization is reviewed. An analysis is then given to computational algorithms in retrieval. Several basic search routines are then defined and shown to envelope all computational tasks. Section III is devoted to the discussion of queuing in terms of the processing time. Sections IV and V are devoted to the analysis of the response time of inverted files and linear files as imbedded in their respective queuing environments.

II. File Organization and Basic Processing Techniques

The most common document retrieval system makes use of coordinate indexing and searching on linear files. Each document is assigned a given number of index terms, usually not more than fifteen per document. On the linear file each record contains information of a document and its index terms.

The queries are formed by choosing a set of index terms. A simple query would require matching of a subset of the terms. This type of operation usually results in an abundance of irrelevant output documents. A better formulation would be to assign as query a Boolean function of the terms, e.g.,

$$(A \vee B \vee C) \wedge (D \vee E \vee F) \wedge (G \vee H).$$

Although the linear file remains the same, processing of a Boolean query would take more time and hopefully yield more satisfactory results. The Boolean query and linear file are the basis of many operational large scale document retrieval systems. Linear files are suited for batch processing and conveniently implemented on magnetic tapes. Their organization, however, hinders response time as the same processing routine is executed whether one query or a hundred queries are presented. In fact, up to a certain point when the system ceases to be input-output limited the response time is essentially constant.

Each record in an inverted file consists of a single index term with the location number of those documents which use this term in their index. When a query is submitted, the inverted file portions of those terms given in the query are retrieved. Further processing based on the linear or Boolean query is done on these inverted file portions to extract the accession number of those documents which meet the requirements. Although the processing of a single query may be reasonably fast, the system is loaded down quickly when the demand increases. A detailed analysis of this will be given in Section IV.

When the number of terminals in the system is small the processing time is negligible as compared with time needed for search in the files. However, in this mode of operation the system's processing capability is not well utilized. When the number of terminals increases the processing time increases rapidly while more records are retrieved from the file. This results in a more favorable situation for the search operation as more overlaps in mechanical motion are possible. Hence the bottleneck in retrieval gradually shifts toward the processor. To accurately analyze the response time it is, therefore, necessary to get a reasonably accurate estimate of the processing time of the search algorithms. What follows is a detailed discussion of the various algorithms and their estimated processing time. Although no attempt is made to show their optimality, these algorithms are believed to be an accurate indicator of the computational complexity of their respective search missions.

A basic search algorithm relates to matching of terms between two sequences of location numbers. Whenever a match is observed a count is made. The document location number is stored whenever the count exceeds a fixed threshold. This algorithm is also useful for initial screening of documents in a two level linear file Boolean search. In this case the threshold is set at one. This threshold is implicitly assumed in the description of the BASIC SEARCH ALGORITHM below.

BASIC SEARCH ALGORITHM:

```
Given:          $a_1$  - - -  $a_x$
                $b_1$  - - -  $b_y$
Initiate:  $i = x$;  $j = y$;
   *    :  $a_i - b_y = 0$;     PRINT; EXIT;
           $a_i - b_j < 0$;
                $j = j-1$;
                    $j=0$;  EXIT;
                    $j \neq 0$;  GO TO *;
           $a_i - b_j > 0$;
                $i = i - 1$;
                $i = 0$;    EXIT;
                $i \neq 0$;    GO TO *;
```

An examination of this typical search algorithm reveals the fact that the number of loops it goes through is bounded by x+y. As the program stops whenever i or j equals zero some saving is possible. We show in the appendix that when this savings is taken into account, the average number of loops θ is given by

$$\theta = x \frac{y}{y+1} + y \frac{x}{x+1}$$

We shall then estimate the computation time of the BASIC SEARCH ALGORITHM to be

$$t_1 = 4\theta u \qquad (1)$$

where 4 is the length of the loop, and u is the average time for processing one instruction.

BASIC BOOLEAN SEARCH ALGORITHM FOR LINEAR FILES

```
Given:   $i = 1, 2,$        $i_o$
         $j(1)\ j(2)$  ----    $j(i_o)$
SEARCH:  $A_{ij}$;
         YES;         GO TO *;
         NO;          $j = j - 1$;
                      $j = 0$;  EXIT;
                      $j \neq 0$;  GO TO SEARCH;
*    :   $i = i - 1$;
             $i \quad 0$;  PRINT:  EXIT:
             $i \neq 0$;  $j = j(i)$;  GO TO SEARCH;
```

The average length of the loop is five plus the processing of SEARCH A_{ij} which is a special case of the BASIC SEARCH ALGORITHM. It is estimated that SEARCH A_{ij} shall take at most $(d+1)4u$ steps where d is the average number of terms per document. Hence the processing time for the BASIC BOOLEAN SEARCH ALGORITHM is

$$t_2 = h[5u + (d+1)4u] \cong 4hu[d+2] \qquad (2)$$

where h is the average number of terms in a request.

Similarly we may construct two basic search algorithms for the inverted file:

BASIC BOOLEAN "OR" ALGORITHM FOR INVERTED FILES

```
Given:     $a_o = b_o = 0$
           $a_1, a_2, - - - , a_x$
           $b_1, b_2, - - - , b_y$
Initiate: $i = x; j = y;$
   *    : $a_i - b_j = 0;$  PRINT  $a_i;$
           $i = i - 1;$
           $i \neq 0;$        GO TO *;
           $i = 0; j \neq 0;$ GO TO *;
                  $j = 0;$ EXIT;
           $a_i - b_j < 0;$ PRINT bj;
            $j = j - 1;$
                  $j \neq 0;$ GO TO *;
                  $j = 0; i \neq 0;$  GO TO *;
                          $i = 0;$  EXIT;
           $a_j - b_j > 0;$ PRINT  $a_i;$
                  $i = i - 1;$
                         $i = 0;$   GO TO *;
                         $i = 0;$   $j \neq 0;$ GO TO *;
                                    $j = 0;$ EXIT;
```

BASIC BOOLEAN "AND" SEARCH ALGORITHM FOR INVERTED FILES

```
Given    :  $a_1\ a_2 - - - a_x$
            $b_1\ b_2 - - - b_y$
Initiate:   $i = x;\quad j = y;$
   *     :  $a_i - b_j = 0;$  PRINT  $a_i;$
                $i = i - 1;$
                    $i = 0;$  EXIT;
                    $i \neq 0;$  GO TO *;
            $a_i - b_j < 0;$
                $j = j - 1;$
                    $j = 0;$ EXIT;
                    $j \neq 0;$ GO TO *;
            $a_i - b_j > 0;$
                $i = i - 1;$
                    $i = 0;$  EXIT;
                    $i \neq 0;$  GO TO *;
```

The processing time of the BASIC BOOLEAN "OR" ALGORITHM can be estimated as

$$t_3 = (2f)(6u) = 12fu \tag{3}$$

where f is the average number of location numbers per index term and the average length of the loop is six. Similarly the processing time of the

BASIC BOOLEAN "AND" ALGORITHM can be estimated as

$$t_4 = (2f)(4u) = 8fu \qquad (4)$$

III. Queuing

The basic system structure considered here consists of a computer with m remote terminals. In a terminal-machine interaction, a user submits a request in the form of a Boolean function of index terms and after a waiting period the machine displays at time T_F the first of a list of documents satisfying the request and at time T_L the last. We will refer to T_F and T_L as the first and last response times. ET_F and ET_L are the expected values of T_F and T_L. Initially we will assume that requests are processed one at a time in order of arrival. When a request is made it is processed immediately unless the computer is busy. In this case the request joins a queue. The average search time α is the time from when a request leaves the queue and begins to be processed until the last requested document is displayed. The average search time obviously depends on detailed system characteristics and file organization. It, as well as ET_F, will be dealt with in Sections IV and V. The purpose of this section is to make some rather general statements about ET_L for a given α.

Suppose that the m terminals operate independently and that the average terminal use time, that is, the time from when a terminal receives the last document satisfying its request until it submits another, is β. The following heuristic argument relates ET_L, α and β. Let W be the average time a request spends in queue. Then the probability that at an arbitrary time a terminal is waiting for a response in $\frac{\alpha + W}{\alpha + \beta + W}$. The fraction of terminals in this condition is $\frac{\alpha + W}{\alpha + \beta + W}\, m$. Thus when a request arrives it has to wait a time

$$W = \frac{\alpha + W}{\alpha + \beta + W}\, m\, \alpha. \qquad (5)$$

Solving for W we find that when both m and $\frac{\alpha m}{\beta}$ are large enough

$$W \approx (m - 1)\, \alpha - \beta \qquad (6)$$

so that

$$ET_L \approx m\, \alpha - \beta \qquad (7)$$

A rigorous treatment of this problem has been given by Takács [1] who has shown that if the terminal use time has the exponential distribution with mean β, then

$$ET_L = m\, \alpha - \beta (1 - P_{m-1}). \qquad (8)$$

Where the factor $1 - P_{m-1}$ is the probability that at the end of a search the queue is not empty. If the minimum possible value of the search time is z, where of course $z \leq \alpha$ then

$$P_{m-1} \leq \left[1 + \sum_{j=1}^{m-1} \binom{m-1}{j} \prod_{k=1}^{j} (e^{kz/\beta} - 1) \right]^{-1} \qquad (9)$$

As Eq.(9) shows, P_{m-1} converges to zero as the number of terminals m increases. For the values of m, z and β in our applications below, we can verify that

$P_{m-1} \approx 0$. Thus we can use Eq.(7) as the relation between ET_L, α and β. When the system has many terminals, the final response time is much longer than the search time. This fact is due to queuing. Let M be the number of requests waiting for service at an arbitrary time. Then from Eq.(7) we may conclude that

$$E\,M \cong (m-1) - \beta/\alpha \tag{10}$$

which shows that long queues are common whenever m is large compared with β/α.

The fact that many requests are waiting with one at a time service leads to a consideration of batch service. Let α_k be the average search time when requests are serviced in batches of k. It is reasonable to expect that with batching we can organize the search efficiently and find that the average time to service a batch of k requests is less than the time to service k requests one at a time. Supposing for now that this is true, that is, that $\alpha_k < k\,\alpha$, we must ask if batching provides any improvement in response time. To answer this question we consider a model like the one at a time model except that now requests are serviced k at a time in an average time α_k. If at the end of a search only p requests are in queue, the system waits until k-p additional requests arrive and then resumes operation Arguing as before we find that when $P_{\frac{m}{k}-1} \approx 0$ we have

$$ET_L \approx \frac{m\,\alpha_k}{k} - \beta . \tag{11}$$

This shows that batch service is better than one at a time service if α_k increases less than linearly with k. In the next two sections we examine the behavior of α_k as a function of k for two types of retrieval.

IV. Inverted File

The inverted file search consists of four basic operations.

1. Determine the locations of the inverted file records corresponding to all terms appearing in a given Boolean request.
2. Access the records.
3. Compare the records and identify the document locations satisfying the search request.
4. Access and display the bibilographic data at all locations found in part 3.

As can be seen the search has two processing and file reading phases. We will assume that either a random addressing or a core-based indexing scheme is used to relate terms to inverted file records. When this is done the time required for the first processing phase, part 1 is negligible compared with that for the rest of the search. The second processing phase, part 3 must be examined in more detail. Let γ be the average processing time for a single request. As before let h be the average number of terms per request and d the average number of terms per document. A request is a Boolean function of its terms. During processing "AND" and "OR" operations are performed on the inverted file records corresponding to the terms in a request. Suppose a request has twice as many "ORs" as "ANDs". We will estimate the time for processing all the "ORs" and one "AND". The remaining

"ANDs" should involve smaller lists. Using the algorithm of Section II the time to do the "ANDs" is $2/3n(8ku) = \frac{16}{3}$ hdu.

After an "AND" operation the combined list will have roughly 2d entries so an "OR" operation will take a time of about $2d(8u) = 16du$. Neglecting the time for "ANDs" with the remaining lists we can estimate the average processing time for our request to be

$$\gamma = 16du[1+h/3] \tag{12}$$

The remaining search time consists of reading the inverted and document files. We will limit the discussion to the case where the two files are on separate disc units. Each unit has several disc modules and each module has an independent read mechanism. We will also suppose that the two disc units are on separate channels. During the search, a file reading period begins either at step 2 or at step 4. In either case the disc controller is presented with a matrix (ℓ_{ij}) of record locations where ℓ_{ij} is the location of the i^{th} record to be read from the j^{th} module. For simplicity we will assume that the same number n of records are read from each module. If the n locations are distributed randomly on a module then the probability that the distance from one disc edge to the first location or between adjacent locations is greater than a, $P_n(a)$ is given by

$$P_n(a) = \frac{M-a}{M} \times \frac{(M-a)u-1}{Mu-1} \times \ldots \times \frac{(M-a)u-n+1}{Mu-n+1} \tag{13}$$

where M is the number of tracks per disc and u is the number of records per cylinder. From $P_n(a)$ and the seek time characteristic for a particular disc unit we can compute the average seek time between locations. When n is small compared with M and u the result is essentially the same as the average distance between n points randomly spaced on an interval $[0,M]$. This average is $\frac{M}{n+1}$. Rather than treat the case where the locations are randomly distributed we will assume that the n locations for a particular module are evenly spaced $\frac{M}{n+1}$ tracks apart. Thus the distance between the first track and the first location or between any two adjacent locations is $\frac{M}{n+1}$. Now suppose there are S disc modules and that nS records are to be read. The following procedure is used. Each read arm starts from one edge of its file and sweeps across to the opposite edge, stopping for record seek and read operations every $(\frac{M}{n+1})$ tracks. Let T(k) be the time for an arm to move k tracks. The S read arms move simultaneously to the first S locations in time $T(\frac{M}{n+1})$. Now this is followed by a waiting period while the correct records rotate into position for reading. For the inverted file we expect the records to be long and perhaps occupy a whole track. If we put several markers around each track then we can begin reading when the first marker appears. The record can be assembled into its correct order in the processor from a knowledge of the markers. If we have enough markers then the waiting time to the first one is negligible and the rotational delay the same as the read time which is the disc rotation time R_o. Thus after the access arms move to the first S locations, one arm say A reads in time R_o while the others wait for the channel. After reading, arm A moves to the next location and the next arm begins to read. The process continues in this way. If $T(\frac{M}{n+1}) \geq (S-1)k_o$, then when arm A finishes its second seek operation, the channel will be free. The same applies to the other arms. In this case the average time to read all the inverted file locations is

$$T_n = nT(\frac{M}{n+1}) + (S-1+n)R_o. \qquad (14)$$

Now if $T(\frac{M}{n+1}) < (S-1)R_o$ then arm A finishes its second seek operation before the other arms have finished reading. In this case all but the first of the track seek times are overlapped by rotational delays and

$$T_n = T(\frac{M}{n+1}) + nSR_o \qquad (15)$$

These two cases can be written together as

$$T_n = T(\frac{M}{n+1}) + nSR_o + (n-1)[T_n-(S-1)R_o]^+ \qquad (16)$$

where

$$\begin{aligned}[x]^+ &= 0 \quad \text{if} \quad x < 0 \\ &= x \quad \text{if} \quad x \geq 0\end{aligned}$$

The first term is the initial track seek time. The second term is the total rotational delay. For an inverted file with the marking scheme outlined above nSR_o is the total read time. The third term is the remaining track seek time after accounting for the overlapping of track seeks with rotational delays.

The situation is essentially the same with the document file. Here we would expect to have several records per track. The average rotational delay is $(\frac{1}{2}+f)R_o = R'$ where f is the fraction of a track occupied by a record. The average time to read all the document file locations is given by Eq.(16) with R' in place of R_o. The rotational delay in document file reading could be decreased by using sector addressing as suggested by Wang and Ghosh [3].

However as we will see in the example below, it is the reading time for the inverted file and not the document file that is the real limitation in search time reduction.

The dependence of Eq.(16) on the batch size k can be made explicit by setting kD = nS where for the inverted file the factor D is the average number of terms per request and for the document file D is the average number of documents satisfying a request. The average lookup time for a batch of k, L(k) is then

$$L(k) = T\left(\frac{M}{\frac{kD}{S}+1}\right) + kDR + \left(\frac{kD}{S} - 1\right)\left[T\left(\frac{M}{\frac{kD}{S}+1}\right) - \left(S - 1\right)R\right]^{+} \qquad (17)$$

where R is R_o or $(\frac{1}{2}+f)R_o$ depending on the file. The utility of batching is in the fact that only the second term of L(k) increases linearly with k. The other two terms decrease as k increases. With batching the part of file lookup time due to track seeks can be made negligible compared with the rotational delay time.

When the inverted and document files are on separate channels parts of the total search can be overlapped. Parts 1-3 of the search can be done for one batch while part 4 is being done for the other batch. Let $L_I(k)$ and $L_D(k)$ be the inverted and document file look-up times. Then the time for parts 1-3 of the search is

$$L_I(k) + k\gamma \qquad (18)$$

where γ is the average processing time estimated above. This figure is conservative since some processing can begin before all the records are accessed.

The average time for an overlapped search α_k is then

$$\alpha_k = L_D(k) + \left[L_I(k) + k\gamma - L_D(k)\right]^+ \qquad (19)$$

since parts 1-3 of a search must be completed before part 4 can begin.

In a strict sense we cannot use α_k in the queuing Eq.(11). This is because there may be times when the queue is empty and there is no batch to overlap with the batch currently being serviced. However when $P_{\frac{m}{k}-1} \approx 0$ the fraction of time that the queue is empty is negligible and we will use Eq.(19) to describe the overlapped search. We have then that

$$ET_L \approx \frac{m}{k}\{L_D(k)+[L_I(k)+k\gamma-L_D(k)]^+\} - \beta \qquad (20)$$

Rather than compute ET_F directly we can notice that it is lower bounded by the queuing time and upper bounded by ET_L so that

$$ET_L - \alpha_k \leq ET_F \leq ET_L. \qquad (21)$$

These bounds are tight when m is large.

<u>Example</u> Consider a collection of $8 \cdot 10^5$ documents with 10^4 terms and an average of 15 terms per document. This gives an average of 1200 documents per term. We will suppose that the average number of terms per request is 8 and the average number of documents satisfying a request is 16.

The inverted and document files are stored on two separate IBM 2314 disc units. These units have 8 modules each. A module has M = 200 cylinders with a rotational time $R_o = 25$ ms. Allowing 150 bytes for a document file entry and 3600 bytes for an inverted file entry the document file will fit on 8 modules with 25 records per track and the inverted file on 2 modules with one record per track. The inverted file is not packed densely since random addressing is used.

Using these numbers we have for the inverted file

$$L_I(k) = T\left(\frac{200}{4k+1}\right) + 200k + (4k-1)\left[T\left(\frac{200}{4k+1}\right) - 25\right]^+ \qquad (22)$$

and for the document file

$$L_D(k) = T\left(\frac{200}{2k+1}\right) + 216k + (2k-1)\left[T\left(\frac{200}{2k+1}\right) - 94.5\right]^+ \qquad (23)$$

From the published table for 2314 seek times we can find $T(\frac{200}{4k+1})$ and $T(\frac{200}{2k+1})$. It turns out that the $[\ ]^+$ term in Eq.(23) is zero for $k \geq 1$. Using the linear approximation for the table we find that

$$\begin{aligned} L_I(1) &= 397 \\ L_I(k) &= 220k + \frac{4k}{4k+1}\,300 + 25, \qquad k \geq 2 \end{aligned} \qquad (24)$$

and

$$\begin{aligned} L_D(k) &= 62 + \frac{187}{2k+1} + 216k, \qquad 1 \leq k \leq 3 \\ &= 30 + \frac{300}{2k+1} + 216k, \qquad k \geq 4 \end{aligned} \qquad (25)$$

From Eq.(17) we can compute the average processing time $\gamma = 94$ ms. So using Eq.(19) we have for the overlapped search time

$$\alpha_k = L_I(k) + k\gamma . \tag{26}$$

For this example at least α_k does not depend on $L_D(k)$ since $L_I(k) + k\gamma > L_D(k)$. To see the effects batching on the response time it is of more interest to compute α_k/k. We have

$$\begin{aligned} \alpha_k/k &= 491 && k = 1 \\ &= 314 + \frac{1200}{4k+1} + \frac{25}{k} && k \geq 2 \end{aligned} \tag{27}$$

As k becomes large α_k/k approaches 314 which is just the sum of the rotational delay and processing time for parts 1-3 of the search. Some values of α_k/k appear in the table.

k	1	2	3	4	5	6	7	8	9	10
α_k/k	491	460	415	391	376	366	359	353	349	346

Using these values of α_k/k we can compute ET_L and plot it as a function of k. The graph shows ET_L curves fo[illegible] [illegible]00, 500 and 100 terminals with β = 20 sec.

V. Linear Files and Round Robin Strategy

Linear files have been used extensively in a batch mode for several reasons. The serial nature of the records makes it economical for sequential processing which minimizes seek time. As random access is not required, it is very attractive for systems with magnetic tapes only.

To evaluate the potential of linear files in an on-line environment, let us first calculate α_k, the time required for processing all documents against k queries. We will find that file data can be read from tapes into core faster than it can be processed. Consequently we will assume that the file portion to be examined is always in core when it is needed. This being the case it is seen that

$$\alpha_k = Ns_k \tag{28}$$

where N is the number of documents in the entire file, and s_k is the time required for processing k requests at once against the terms of one document. With a two level search, the first step is to compare all the kh terms of the k queries with the d terms of a document. If there are any matches an exact comparison is made at the second level. From considerations in Section II it is seen that

$$s_k = 4\theta_k u + kp_1 \; t_2 \tag{29}$$

where

$$\theta_k = kh \frac{d}{d+1} + d \frac{kh}{kh+1} \tag{30}$$

p_1 is the average number of queries passing the first level search, and t_2 is the processing time of a single document with one query. The value of t_2 has been estimated in of (2) as

$$t_2 = 4hu[d+2] \tag{31}$$

Since p_1 is usually a small number

$$s_k \approx 4\theta_k u \tag{32}$$

From Eq.(11) we see that for batches of k

$$ET_L = \frac{m\alpha k}{k} - \beta = \frac{4mNu\theta_k}{k} - \beta = 4mNu\left[h\frac{d}{d+1} + d\frac{h}{kh+1}\right] - \beta \tag{33}$$

From these formulas it is clear that conventional systems based on the linear file give a long last response time, although batch processing does improve the performance somewhat.

However the documents satisfying a search request can be read out as soon as they are identified. For a particular request the average time to the first document is $\frac{\alpha k}{d}$ so we have

$$\begin{aligned} ET_F &= ET_L - \alpha_k + \frac{\alpha k}{d} \\ &= \alpha_k\left(\frac{m}{k} + \frac{1}{d} - 1\right) \end{aligned} \tag{34}$$

As noted earlier we cannot use Eq.(11) for large values of k. However for large k and hence large α_k, we can argue that β should be small and Eq.(11) holds for $\beta = 0$. This fact is used to obtain Eq. 33. The best value of k in Eq.(33) is k = m which gives

$$ET_F = \frac{\alpha_M}{d} = \frac{4Nu}{d}\left[mh\,\frac{d}{d+1} + d\,\frac{mh}{mh+1}\right] \qquad (35)$$

$$\approx \frac{4Numh}{d}$$

Our conclusion is that processing should be done in such a way that new queries enter into processing as soon as possible and do not queue for machine time, that is, the batch size is m. Increasing the batch size makes an almost negligible decrease in ET_L but does improve ET_F.

Example Consider a system with $m = 100$, $d = 15$, $h = 10$, $N = 8\cdot10^5$, $u = 0.5\cdot10^{-6}$ and then we have for $k = m$,

$$ET_L = 25 \text{ min}$$

$$ET_F = 1.7 \text{ min}$$

The graph of Fig. 2 shows ET_L and ET_F as a function of batch size.

VI. Concluding Remarks

The response time for on-line document retrieval systems has been investigated. It is shown that for systems with a large number of terminals the response time is approximately linear with m, the number of terminals. Two file organizations have been evaluated. It is found that if traffic is not too heavy the inverted file seems adequate. The linear file is rather slow in comparison. The general conclusion here is that conventional techniques for document retrieval are not adequate for on-line systems when the number of terminals is very large. For such systems to be functional one needs to develop new and original file organization and search techniques.

We wish to note that when a parallel processor such as the ILLIAC IV is available the efficiency of the system can improve by a factor at least equal to the number of PU's available. For sixty-four parallel PU's the system can handle about one hundred times the load. The additional improvement is obtained because of savings in execution time.

References

[1] Cuadra, C. A., "Annual Review of Information Science and Technology, Vol. 4, 1969, Brittannica.

[2] Takács, L., Introduction to the Theory of Queues, Oxford University Press, Oxford, 1962.

[3] Wang, C. P. and Ghosh, S. P., "Analysis and Modeling of a Multimodule Disc Storage System with Sector Scheduling," Report, IBM San Jose Research Laboratory.

[4] Abate, J., Dubner, H., and Weinberg, S. B., "Queuing Analysis of the IBM 2314 Disc Storage Facility," Journal ACM, Vol. 15, No. 4, pp. 557-589, October, 1968.

[5] Feller, W., "An Introduction to Probability and Its Applications," Vol. 1, John Wiley and Sons, 1957.

[6] Barnes, G. H., Brown, R. M., Kato, M., Kuck, D. J., Slotnick, D. L., and Stokes, R. A., "The ILLIAC IV Computer," IEEE Trans. Computers, Vol. C-17, pp. 746-757, Aug., 1968.

Appendix

Theorem: Suppose we are given two sequences of points $a_1, a_2, --- a_x$ and $b_1, b_2, --- b_y$ obtained by ordering x and y points each uniformly distributed between 0 and M. Then if the basic search algorithm is used to match the sequences the average number of steps is given by

$$x(\frac{y}{y+1}) + y(\frac{x}{x+1})$$

Proof of Theorem

From the probability that $a_x \leq t$ the density function for a_x is found to be

$$(\frac{t}{M})^{x-1}(\frac{x}{M}).$$

Similarly the density for b_y is

$$(\frac{s}{M})^{y-1}(\frac{y}{M})$$

The average number of processing steps when $t > s$ is

$$y + (x-1)\frac{s}{t}$$

It is

$$x + (y-1)\frac{t}{s}$$

when

$$s > t$$

The expected number of processing steps is, therefore, given by

$$\theta = \int_0^M (\frac{t}{M})^{x-1} (\frac{x}{M}) [\int_0^t (\frac{s}{M})^{y-1} (\frac{y}{M}) (y + \frac{(x-1)s}{t}) ds$$

$$+ \int_t^M (\frac{s}{M})^{y-1} (\frac{y}{M}) (x + \frac{(y-1)t}{s}) ds] dt$$

It is seen that

$$\int_0^t (\frac{s}{M})^{y-1} (\frac{y}{M}) (y + \frac{(x-1)s}{t}) ds$$

$$= \frac{y(x+y)}{(y+1)} \frac{t^y}{M^y}$$

$$\int_t^M (\frac{s}{M})^{y-1} (\frac{y}{M}) (x + \frac{(y-1)t}{s}) ds$$

$$= x + \frac{yt}{M} - \frac{t^y}{M^y} (x+y)$$

Therefore

$$\theta = \int_0^M (\frac{t}{M})^{x-1} (\frac{x}{M}) [x + \frac{yt}{M} + \frac{t^y}{M^y} (x+y) (\frac{y}{y+1} - 1)] dt$$

$$= y(\frac{x}{x+1}) + x(\frac{y}{y+1})$$

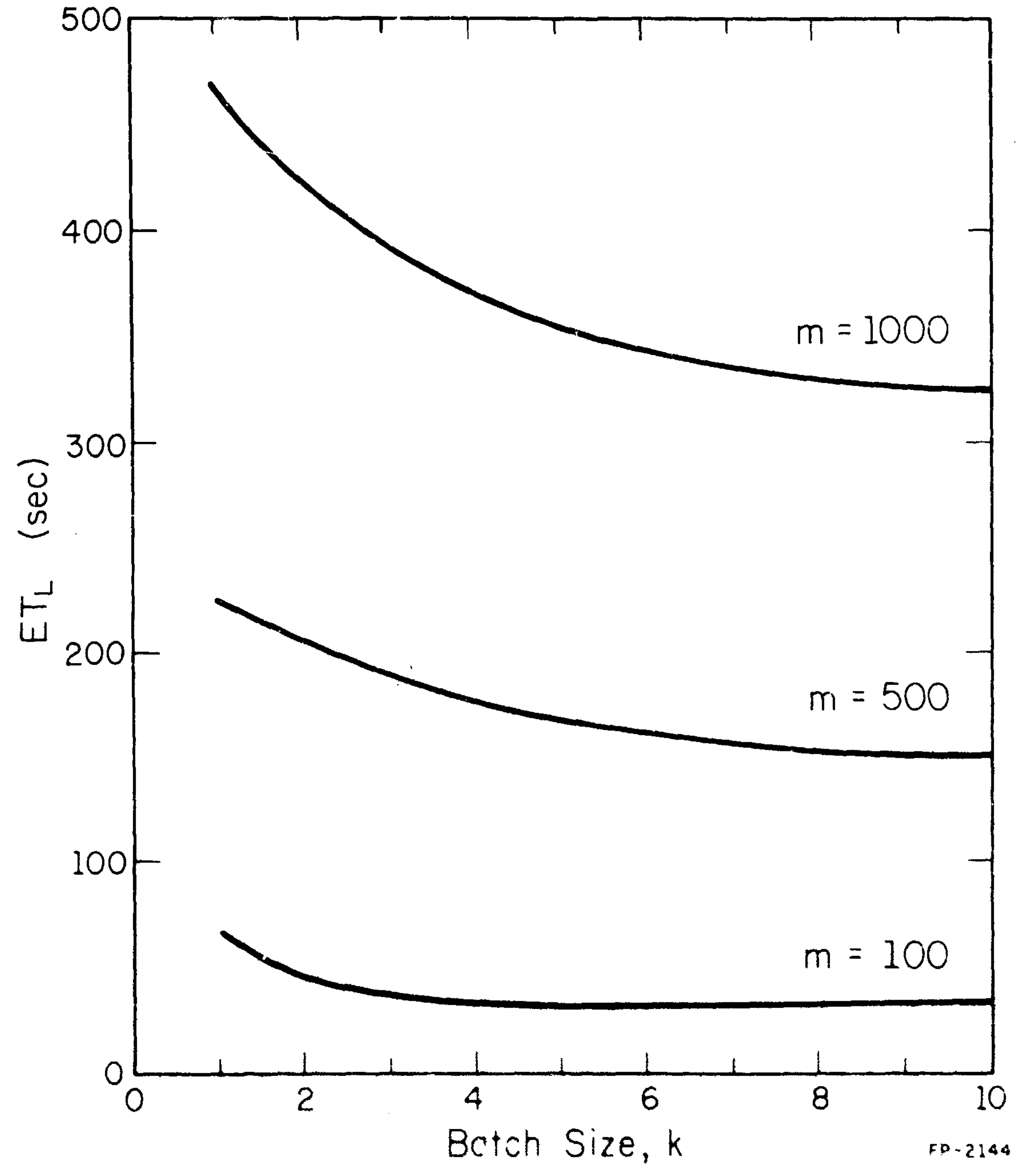


Fig. 1 Average response time vs batch size for the inverted file.

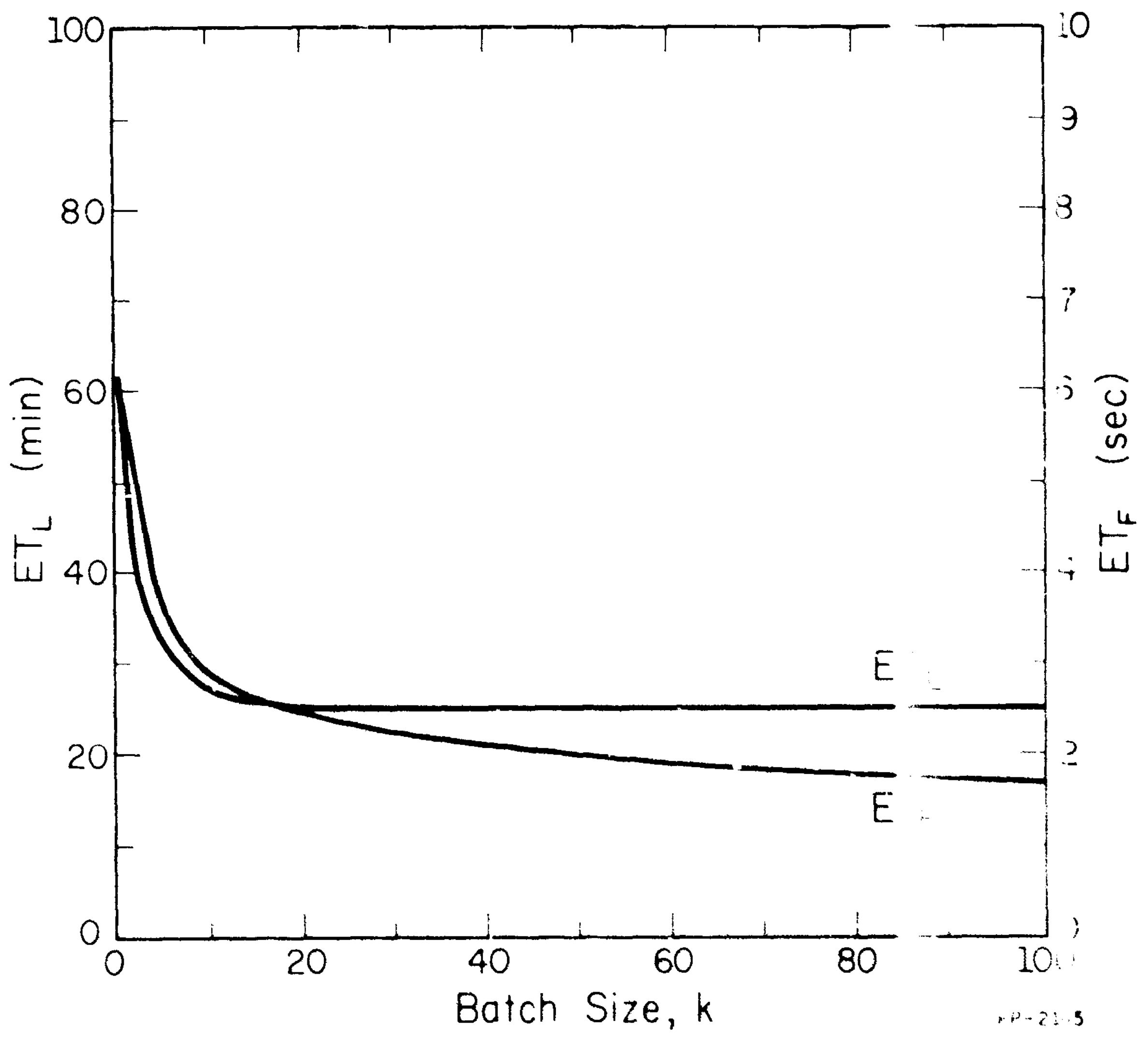
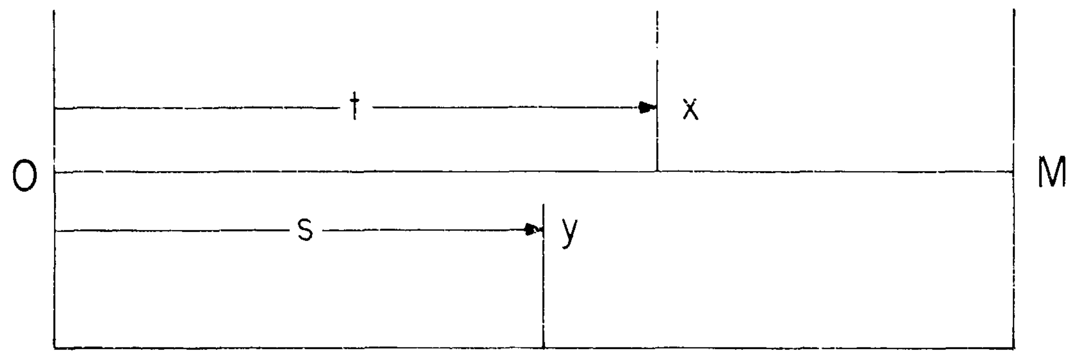


Fig. 2 Average response time vs batch size for the linear file



FP-2146

Fig. 3 Distribution of points.

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* University of Illinois Coordinated Science Laboratory Urbana, Illinois 61801	2a. REPORT SECURITY CLASSIFICATION 2b. GROUP

3. REPORT TITLE

DESIGN CONSIDERATIONS OF ON-LINE DOCUMENT RETRIEVAL SYSTEMS

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*

CORDARO, J.T. & CHIEN, R.T.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
February, 1970	28	6

8a. CONTRACT OR GRANT NO. DAAB-07-67-C-0199; also in part NSF Grant GK-2339 and OE C-1-7-071213-4557. b. PROJECT NO. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) R-456 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* UILU-ENG 70-201

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Joint Services Electronics Program thru U. S. Army Electronics Command Fort Monmouth, New Jersey 07703.

13. ABSTRACT

Response time of on-line document retrieval systems are analyzed. Linear and inverted file organizations are considered and their response times are evulated.

DD FORM 1 NOV 65 1473

Security Classification

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Document Retrieval						
Inverted Files						
Linear Files						

Security Classification